Windows Standard

Serial Communications

for C/C++

Programmer's Manual

(WSC_4C)

Version 5.4

August 11, 2015.

This software is provided as-is.
There are no warranties, expressed or implied.

Copyright (C) 2015
All rights reserved

MarshallSoft Computing, Inc.
Post Office Box 4543
Huntsville AL 35815

Email: info@marshallsoft.com
Web: www.marshallsoft.com

MARSHALLSOFT is a registered trademark of MarshallSoft Computing.

TABLE OF CONTENTS

1 Introduction Page 3
  1.1 Features Page 4
  1.2 Documentation Set Page 6
  1.3 Example Program Page 6
  1.4 Installation Page 7
  1.5 Uninstalling Page 7
  1.6 Pricing Page 7
  1.7 Updates Page 7
2 Library Overview Page 8
  2.1 Using the Library Page 8
  2.2 Keycode (License Key) Page 8
  2.3 Console Mode Page 8
  2.4 Limitations on COM Ports Page 9
  2.5 Static Linking Page 9
  2.6 Using Threads Page 10
  2.7 Calling WSC Functions from C++ Page 10
  2.8 Adding WSC Functions to an Existing Program Page 10
  2.9 SioEvent Logic Page 10
  2.10 Virtual Serial Ports Page 10
  2.11 Explicitly Loading the WSC DLL Page 11
  2.12 Targeting a 64-Bit CPU Page 11
3 Supported Compilers Page 13
  3.1 Microsoft Visual C++ Page 13
  3.2 Microsoft Visual C# (C-Sharp) Page 15
  3.3 Microsoft C/C++ (16-bit) Page 15
  3.4 Borland C++ Page 16
  3.5 Turbo C/C++ for Windows Page 16
  3.6 Borland C++ Builder Page 17
  3.7 Watcom C/C++ Page 17
  3.8 LCC C Page 18
  3.9 MinGW C Page 18
  3.10 Digital Mars Page 18
4 Compiling Programs Page 19
  4.1 Compiling WSC Source Code Page 19
  4.2 Compiling Example Programs Page 20
5 Example Programs Page 22
6 Additional Examples Page 33
7 Revision History Page 34

1 Introduction

The **Windows Standard Serial Communications Library for C/C++ (WSC4C)** is a toolkit that allows software developers to quickly develop serial communication applications in C/C++, Visual C++, .NET and Visual C#.

The **Windows Standard Serial Communications Library** (WSC) is a component DLL library used to create serial communications programs that access data from a serial port using RS232 or multi-drop RS422 or RS485 ports. WSC also supports virtual serial ports such as Bluetooth serial and USB to serial converters. The **WSC** component library uses the Windows API for all communication and can be used to easily write applications to control serial devices such as barcode scanners, modems, lab instruments, medical devices, USB serial devices, scales, GPS navigation, etc.

This **Windows Standard Serial Communications Programmer's Manual** provides information needed to compile programs in a C/C++/C# programming environment.

WSC4C includes more than 25 C/C++ example programs demonstrating serial port communication functions. Microsoft Visual Studio, Microsoft Foundation Class (MFC), Borland C++ Builder (BCB). Microsoft Visual C++ .NET and Visual C# (C-sharp) examples are included.

The **Windows Serial Communications Library for C/C++ (WSC4C)** supports and has been tested with C/C++, Microsoft Visual C++, Visual Studio .NET Framework, Visual C++ Express, Visual C#, Borland C/C++, Borland Turbo C++ for Windows, Borland C++ Builder, Watcom C/C++, MinGW C++, Digital Mars C, and LCC C compilers. **WSC4C** can also be used with most other C/C++ Windows compilers.

The **Windows Standard Communications Library SDK** includes Win64 and Win32 DLLs (WSC64.DLL and WSC32.DLL). The DLLs can also be used from any language (Visual Basic, VB.NET, ACCESS, EXCEL, VBA, Borland Delphi, Visual FoxPro, COBOL, Xbase++, Visual dBase, etc.) capable of calling the Windows API. **WSC4C** runs under all versions of Windows through Windows 10. Evaluation versions may be downloaded from our web site at

http://www.marshallsoft.com/serial-communication-library.htm

Our goal is to provide a robust serial communication component library that you and your customers can depend upon. A fully functional evaluation version is available. Contact us if you have any questions.

When comparing the **Windows Standard Serial Comm Library** against our competition, note that:

1. WSC4C is a standard Windows DLL (NOT an OCX or ActiveX control) and is much smaller than a comparable OCX or ActiveX control.
2. Win64 and Win32 are included.
3. WSC4C does NOT depend on ActiveX or Microsoft Foundation Class (MFC) libraries or similar "support" libraries.
4. The WIN32 / WSC64 versions of WSC are fully threadable.
5. The WSC functions can be called from applications not capable of using controls.

MarshallSoft also has versions of the **Windows Standard Serial Communications Library** for Visual Basic (WSC4VB), Delphi (WSC4D), PowerBASIC (WSC4PB), Visual FoxPro (WSC4FP), Visual dBASE (WSC4DB), and Xbase++ (WSC4XB). All versions of WSC use the same DLLs (WSC64.DLL and WSC32.DLL). However, the examples provided for each version are written for the specified computer programming language.

The latest versions of the **Windows Standard Serial Communications Library (WSC)** can be downloaded from our web site at

http://www.marshallsoft.com/serial-communication-library.htm

1.1 Features

Some of the many features of the **Windows Serial Communications Library** are:

- Comes with 64-bit and 32-bit DLLs.
- Supports RS232, and multidrop RS422 & RS485 ports.
- Can control any serial device connected to a serial port.
- Can be used from GUI mode or console mode programs.
- Can control up to 256 ports simultaneously.
- Can be used with virtual serial ports using Bluetooth serial or a USB to serial converter.
- Includes 46 functions plus modem control.
- Comes with ANSI emulation and ASCII, XMODEM and YMODEM protocols.
- Supports hardware and software flow control.
- Supports for any baud rate (32-bit & 64-bit versions).
- Ability to specify the parity, word size, and number of stop bits.
- Supports binary and text data transfer.
- Is fully thread safe.
- Can send Windows messages on completion of events (incoming character, etc.)
- Supports all versions of Windows through Windows 10.
- License covers all programming languages.
- Royalty free distribution with a compiled application. There are no run time fees.
- Evaluation versions are fully functional. No unlock code is required.
- Can be used from GUI mode or console mode programs.
- Is fully thread safe.
- Implemented as a **standard** Windows DLL, which will work with all versions of Windows.
- Is native Windows code but can also be called from managed code.
- Will run on machines with or without .NET installed.
- Works with all versions of Microsoft Visual C/C++ (v1.52 through Visual Studio 2015).
- Works with Microsoft Visual Studio .NET and Visual C#
- Works with Borland C/C++ (v5.0, v5.5, and Borland C++ Builder).
- Also works with Microsoft Foundation Class, Watcom v11, MinGW, Digital Mars, and LCC.
- Does **not** depend on support libraries. Makes calls to core Windows API functions only.
- Can be used with any program (in any language) capable of calling Windows API functions such as Visual Basic, VB.NET, Visual FoxPro, Delphi, Xbase++, dBASE, COBOL, Access or Excel.
- Can be purchased with (or without) source code.
- Free updates for one year.
- License includes technical support for one year.
- Documentation online as well as in printable format.

A selection of C/C++ example programs with full source code is included. Refer to Section 6 for more details on each of the example programs.

[PROGRAM]	[DESCRIPTION]
BCB	Borland C++ Builder (32-bit C++) example.
CALLBACK	Demonstrates how to implement a WSC callback function.
CONSOLE	A Win32 console mode program similar to SIMPLE.
CS_AT_OK	C# program that transmits "AT" and expects "OK" back.
CS_VERS	C# program that displays the WSC version number.
DEVICE	Program to communicate with serial device.
DIAL	Dials up host (or BBS).
ECHOPORT	Multi-threaded console mode program echoes all input.
EVENT	Blocking version of SIMPLE that uses SioEvent.
EventC	Blocking version of SIMPLE that uses SioEventChar.
EventW	Blocking version of SIMPLE that uses SioEventWait.
FINDER	Finds a modem connected to one of your serial ports.
HELLO	Example program which uses the C++ fSio class.
HOST	Simple one-line host program.
LISTER	Lists all serial ports.
LOADLIB	Loads WSC32.DLL / WSC64.DLL dynamically.
MFC_PGM	Microsoft Foundation Class (MFC) C++ example program.
RS485	RS485 version of CONSOLE.
SELFTEST	Performs COM port functionality testing.
SIMPLE	A simple terminal emulator.
TERM	Terminal emulator with XMODEM, YMODEM, and ANSI support.
VC_CONSOLE	Visual Studio version of the console.c program
VC_DEVICE	Visual Studio version of the device.c program
VC_EVENT	Visual Studio version of the event.c program.
VC_FINDER	Visual Studio version of the finder.c program.
VC_LISTER	Visual Studio version of the lister.c program.
VC_SIMPLE	Visual Studio version of the simple.c terminal.
VC_VERSION	Visual Studio version of the wscver.c (version) program
WSCVER	Program that displays the WSC version number.
XMS, XMR	Console mode XMODEM programs.
YMS, YMR	Console mode YMODEM programs.

1.2 Documentation Set

The complete set of documentation consists of four manuals. This is the first manual (WSC_4C) in the set.

- WSC_4C Programmer's Manual (WSC_4C.PDF)
- WSC User's Manual (WSC_USR.PDF)
- WSC Reference Manual (WSC_REF.PDF)
- SERIAL User's Manual (SERIAL.PDF)

The WSC_4C Programmer's Manual is the language specific (C/C++) manual and provides information needed to install and compile example programs in a C/C++ environment. Read this manual first.

The WSC User's Manual (WSC_USR) discusses language independent serial communications programming issues including modem control. Purchasing and license information is also provided.

The WSC Reference Manual (WSC_REF) contains details on each individual WSC function.

The Serial Communications Manual (SERIAL) contains background information on serial port hardware.

All manuals can be viewed online at http://www.marshallsoft.com/wsc4c.htm

1.3 Example Program

The following example demonstrates some of the **Windows Serial Communications** library functions.

```
#include <windows.h>
#include <stdio.h>
#include "wsc.h"
void main(void)
{int  Code;
 SioKeyCode(0); // 0 is evaluation version key code
 // reset (open) COM1
 Code =  SioReset(COM1,1024,1024);
 if(Code<0)
   {char Temp[80];
    // display error message
    SioWinError((char *)Temp,80);
    printf("ERROR %d : %s\n",Code,(LPSTR)Temp);
    SioDone(COM1);
    exit(1);
   }
 // set the DTR line
 printf("Setting DTR\n");
 SioDTR(COM1,'S');
 // wait for user to quit
 printf("Type any character to exit.\n");
 getch();
 // close port
 SioDone(COM1);
}
```

Refer to section 6.0 for complete examples with source.

1.4 Installation

(1) Before installation of WSC4C, a Windows C/C++ compiler should already be installed and tested.

(2) Unzip WSC4C54.ZIP (evaluation version) or WSCxxxxx.ZIP (where xxxxx is the developer's Customer ID) using any Windows unzip program.

(3) Run the installation program SETUP.EXE which will install all WSC4C files, including copying WSC64.DLL and WSC32.DLL to the Windows directory. Note that no DLL registration is required.

1.5 Uninstalling

Uninstalling WSC4C is very easy. First, delete the WSC project directory created when installing WSC4C. Second, delete WSC64.DLL and WSC32.DLL from the Windows directory, normally C:\WINDOWS. That's it!

1.6 Pricing

A developer license for the **Windows Standard Serial Communications Library** (**WSC4C)** can be purchased for $115 (or $195 with ANSI C source code to the library DLL's). Purchasing details can be found in the WSC User's Manual (WSC_USR), Section_1.3, "How to Purchase" (http://www.marshallsoft.com/wsc_4c.pdf).

Also See INVOICE.TXT or http://www.marshallsoft.com/order.htm

1.7 Updates

When a developer license is purchased for WSC4C, the developer will receive a new set of registered DLLs plus a license file (WSCxxxx.LIC). The license file is used to download updates to the registered DLL's for a period of one year from purchase. Updates can be downloaded from

http://www.marshallsoft.com/update.htm

After one year, your license must be updated to be able to download updates. A developer license can be updated for:

- $30 if the update is ordered within one year of the original purchase or previous update.
- $55 if the update is ordered between one and three years of the original purchase or previous update.
- $75 if the update is ordered after three years of the original purchase or previous update.

The update price includes technical support for an additional year. If source code was previously purchased, updates to the source code can be purchased for $40 along with the DLL update. Note that the registered WSC DLLs never expire.

2 Library Overview

2.1 Using the WSC Library

The **Windows Standard Communications Library for C/C++** has been tested on multiple computers running Windows 95/98/Me/XP/2003/2012/Vista/Windows 7/Windows 8/Windows10 and Windows NT/2000.

The WSC4C library has been tested with several C/C++ compilers, including Microsoft Visual C++ (all versions including Visual Studio .NET and Visual Studio C#), Borland C/C++, Borland C++ Builder, Turbo C/C++ for Windows, and Watcom C/C++.

The SETUP installation program will copy the Lib's and DLL to the Windows directory. Refer to Section 1.4 "Installation". After SETUP is run, the WSC4C files are copied to the directory specified (default \WSC4C). Three sub-directories are created, as follows:

```
DOCS - All documentation files
APPS - All example code
DLLS - All DLLs
```

2.2 Keycode (License Key)

The WSC DLLs each has a keycode encoded within them. The keycode is a 9 or 10 digit decimal number (unless it is 0), and will be found in the file KEYCODE.H. The keycode for the evaluation version is 0. The developer will receive a new keycode and a new set of WSC DLL's after purchasing a license. The KEYCODE is passed to **SioKeyCode**.

If an error message (value -74) is returned when calling **SioKeyCode**, it means that the keycode in the WSC application does not match the keycode in the DLL. After purchasing a license, it is best to remove the evaluation versions of the WSC64.DLL and WSC32.DLL from the Windows search path or delete them before installing the purchased version.

2.3 Console Mode

WSC4C functions can be called from Win32/Win64 console mode programs as well as GUI (Graphical User Interface) programs.

A "console mode" program is a Windows Win32/Win64 command line program running in a command window. Although console mode programs look like DOS programs, they are Win32/Win64 programs that have access to the Win32/Win64 API and the entire Windows address space. Programming using console mode programs reduces the complexity of using GUI code. All console mode programs can be converted to GUI mode by adding the necessary Windows interface code

2.4 Limitations on COM Ports

The 32-bit and 64-bit versions of WSC4C (WSC32.DLL and WSC64.DLL) can use any port from COM1 to COM256, provided that the port is known to Windows and there is physical hardware present.

2.5 Static Linking

The registered user can statically link WSC so that the DLL is not needed. Static linking requires the object file (WSC32.OBJ or WSC64.OBJ) version of WSC (included in the registered package) or source code. For example, to create an application that links WSC32.OBJ statically:

Make sure that all application code that includes WSC.H must define STATIC_LIBRARY before including WSC.H

If using Microsoft Developer Studio, make these changes:

(1) To the project file: Do NOT add WSC32.LIB to your project file.
(2) To the settings: (See "Build Settings" or "Project/Settings")
(3) C/C++ Tab: Add STATIC_LIBRARY to "preprocessor definitions:".
(4) Link Tab: Add WSC32.OBJ to "object/library modules:".
(5) Add #include "wsc.h" to all source files that make calls to WSC functions.

Alternatively, WSC.C can be edited so that they can be compiled and linked like any other program file. In order to do this, remove all code from WSC.C (provided when the source code is purchased) from

```
#ifndef STATIC_LIBRARY
```

through the following

```
#endif
```

2.6 Using Threads

WSC4C is thread safe. Refer to the ECHOPORT.C example program, which demonstrates the use of multiple threads. ECHOPORT also demonstrates the use of the Win32/Win64 Sleep() function.

Also examine the EVENT.C example program, which uses both threads and the **SioEvent** function.

2.7 Calling WSC functions from C++

Like Windows itself, WSC functions are coded in ANSI C, but they can be called directly from both ANSI C programs and from C++ and C# programs.

WSC functions can also be called using the C++ class wrapper **fSio**. Refer to HELLO.CPP for an example.

2.8 Adding WSC Functions to an Existing Program

In order to call WSC functions from an existing program, (1) add

```
#include "wsc.h"
```

to your application source code, (2) link with WSC32.LIB (for MSVC), WSC32BCB.LIB (Borland C/C++ and C++ Builder), WSC32.LIB (Watcom), or WSC32LCC (Win32/LCC), and recompile from source.

For Win64 applications, link with WSC64.LIB rather then WSC32.LIB.

2.9 SioEvent Logic

SioEvent, **SioEventChar**, and **SioEventWait** will block until the specified event occurs. If a call to **SioEvent**, **SioEventChar**, or **SioEventWait** is placed in a thread, then the thread will block but the application calling the thread will not.

- **SioEvent** blocks until the specified event occurs.
- **SioEventWait** blocks until the specified event occurs or the timeout period expires.
- **SioEventChar** blocks until the specified character is received or the timeout period expires.

See the VC_Event.cpp, Event.c, EventChar.c, and EventWait.c example programs.

2.10 Virtual Serial Ports

A "virtual" serial port is COM port that appears to be a real RS232 serial port to the Windows API (and thus to WSC), but is in reality a COM port emulator. Common examples of virtual COM ports are Bluetooth serial ports and USB/serial converter ports.

More information about virtual serial ports can be found in WSC User's Manual (WSC_USR), Section 2.13 (http://www.marshallsoft.com/wsc_usr.pdf**).**

2.11 Explicitly Loading the WSC DLL

When an application program runs that makes calls to WSC32.DLL (or WSC64), the Windows operating system will locate WSC32.DLL (or WSC64.DLL) by searching the directories as specified by the Windows search path. If the WSC DLL (WSC64.DLL or WSC32.DLL) is placed in the \WINDOWS directory (or \WINNT for Windows NT/2000), it will always be found by Windows.

However, WSC32.DLL (or WSC64.DLL) can also be loaded from a specified directory by using the GetProcAddress Win32 (or Win64) API function. For an example, refer to the LoadLib.c program.

2.12 Targeting a 64-Bit CPU

If a compiler generates 32-bit application code and is running on a 64-bit version of Windows, then compiling and linking is the same as it were on a 32-bit Windows system. The 32-bit application code generated will be executed by the Windows WOW64 (Windows on Windows 64-bit) component.

If a compiler generates 64-bit application code and is running on a 64-bit version of Windows, then the compiler must be reconfigured to generate 32-bit application code if the application will call 32-bit DLL's such as WSC32.DLL. The 32-bit application code generated will be executed by the Windows WOW64 (Windows on Windows 64-bit) component.

2.12.1 Visual Studio C++: Versions 2005 through 2015

With a project selected in Solution Explorer, on the Project menu, click Properties.
Click the "Configuration Manager" button in upper right corner.
Click the drop-down button below "Platform".
Click <New...>, then choose "x86" (Win32).

2.12.2 Visual Studio Visual Basic: Versions 2005 through 2015

With a project selected in Solution Explorer, on the Project menu, click Properties.
Click "Build", then "Configuration Manager".
Click the drop-down button below "Active Solution Platform".
Click <New...>, then change "Any CPU" to "x86".

2.12.3 64-bit Application Programs and WSC

64-bit DLLs (such as WSC64.DLL) may only be used by 64-bit application programs running on 64-bit Windows computers. This means that 64-bit application programs must be linked with WSC64.LIB instead of WSC32.LIB.

There are several 64-bit example programs. 64-bit Visual Studio 2008 /2010/2012/2013 project files. Unzip vs2008(64bit)vcproj.zip for VS2008 project files, vs2010(64bit)vcproj.zip for VS2010 project files, vs2012(64bit)vcproj.zip for VS2012 project files or vs2013(64bit)vcproj.zip for VS2013 project files.

```
VS2008:   vc_console.vcproj    vc_device.vcproj     vc_event.vcproj
          vc_finder.vcproj     vc_simple.vcproj     vc_version.vcproj
          vc_xmr.vcproj        vc_xms.vcproj        vc_lister.vcproj

VS2010:   vc_console.vcxproj   vc_device.vcxproj    vc_event.vcxproj
          vc_finder.vcxproj    vc_simple.vcxproj    vc_version.vcxproj
          vc_xmr.vcxproj       vc_xms.vcxproj       vc_lister.vcxproj

VS2012:   vc_console.vcxproj   vc_device.vcxproj    vc_event.vcxproj
& 2013    vc_finder.vcxproj    vc_simple.vcxproj    vc_version.vcxproj
          vc_xmr.vcxproj       vc_xms.vcxproj       vc_lister.vcxproj
```

However, if a compiler generates 32-bit code, the application must be linked with WSC32.LIB even though it may be running on a 64-bit machine.

3.0 Supported Compilers

The **Windows Standard Serial Comm Library** (**WSC4C)** has been tested and works with all versions of the following compilers:

- Microsoft Visual C++
- Visual C++ .Net through Visual Studio 2015
- Visual C#
- Borland C/C++
- Borland C++ Builder
- Borland Turbo C/C++
- Watcom C/C++
- MinGW C++
- LCC
- Digital Mars

Other Windows C/C++ compilers may work as well. Refer also to Section 4.0, "Compiling Example Programs".

3.1 Microsoft Visual C++ (all versions)

Microsoft Visual C++ programs can be compiled from either the command line or from within the Microsoft Visual Studio development environment.

3.1.1 Microsoft Command Line Makefiles

Some of the example programs can be compiled using command line makefiles. All Microsoft Win32 command line makefiles end with '32._m_". To compile using a makefile, use the Microsoft NMAKE utility. For example,

```
NMAKE -f SIMPLE(LIB)32._M_
```

WSC can be used with Microsoft Foundation Class (MFC) programs. Use the MFCPGM32.MAK makefile to compile the MFC_PGM example program.

```
NMAKE -f MFCPGM32.MAK
```

The file *Makefiles(Microsoft).zip* contains the Microsoft Visual C/C++ (4.0, 5.0, 6.0) and Visual Studio (through VS 2015)command line makefiles for most of the example programs.

Project files for Microsoft C/C++ (4.0 thru 6.0) have extensions *.mak and *.dsp

3.1.2 Microsoft Developer Studio (VC v4.0)

To open an existing project, choose "File", then "Open Workspace", and then select "makefiles" from the list of file types. Most of the example programs have Microsoft Developer C/C++ project makefiles, ending with ".MAK" , such as WSCVER.MAK

To create a new project in MSVC v4.0, choose "File", then "New", then "Project Workspace". Select "Application" or "Console Application" for "Type:" and the project name for "Name:". Choose Win32 for platform. Then select "Create". Select "Insert", then "Files into Project". Add all filenames including any resource file (.RC) and WSC32.LIB. Lastly, select "Build", then "Rebuild All".

Be sure to specify /YX rather than /Yu in the project settings [Build, Settings…, C/C++].

3.1.3 Microsoft Visual Studio (VC v5.0)

To open an existing project, choose "File", then "Open Workspace", and then select "makefiles" from the list of file types. Most of the example programs have Microsoft Developer C/C++ project makefiles, ending with ".MAK", such as WSCVER.MAK.

To create a new project in MSVC v5.0, choose "File", then "New", then "Win32 Application" or "Win32 Console Application" and the project name. Check "Create new workspace". Select "Project", then "Add to Project". Add all filenames including any resource file (.RC) and WSC32.LIB. Lastly, select "Rebuild All".

If the compiler complains that it cannot find "_main", "Console Application" was chosen but the program being compiled is a GUI application. If the compiler complains that it cannot find "WinMain", "Application" was chosen but the program being compiled is a Console Mode application. Be sure to specify /YX rather than /Yu in the project settings [Build, Settings…, C/C++].

3.1.4 Microsoft Visual Studio (VC v6.0)

To open an existing project, follow the same directions as for MSVC v5.0, except that a DSP project file may be used instead of the MAK project makefile.

To create a new project in MSVC v6.0, follow the same directions as for MSVC v5.0 above.

3.1.5 Microsoft Visual Studio 2003 through 2015

Open the VC project file (files ending in ".vcproj" or ".dsp"), build and run, as in previous versions of Visual Studio.

Unzip **`vs2008(32bit)vcproj.zip`** for all VS2008 project files.
Unzip **`vs2008(64bit)vcproj.zip`** for all VS2008 project files.

Unzip **`vs2010(32bit)vcxproj.zip`** for all VS2010 project files.
Unzip **`vs2010(64bit)vcxproj.zip`** for all VS2010 project files.

Unzip **`vs2012(32bit)vcxproj.zip`** for all VS2012 project files.
Unzip **`vs2012(64bit)vcxproj.zip`** for all VS2012 project files.

Unzip **`vs2013(32bit)vcxproj.zip`** for all VS2013 project files.
Unzip **`vs2013(64bit)vcxproj.zip`** for all VS2013 project files.

Unzip **`vs2015(32bit)vcxproj.zip`** for all VS2015 project files.
Unzip **`vs2015(64bit)vcxproj.zip`** for all VS2015 project files.

Also refer to Section 4.2 below.

3.1.6 Microsoft Visual C++ Express

The "Express Edition" of Microsoft C++ is available as a free download at
http://www.microsoft.com/express/download/

Open the VC project file (files ending in ".vcproj" or ".dsp"), build and run, as in previous versions of Visual Studio. Also see Section 4.2 below.

3.2 Microsoft Visual C# (C-Sharp)

WSC functions can be called from Visual C# (C-sharp) in the same manner as Win32 API functions.

All Visual C# projects end with extension ".`csproj`". For example,

```
cs_vers.csproj
```

In order to open an existing Visual C# project, choose "File", "Open", and then "Project" from the Microsoft C# Development Environment.. Specify the directory containing the Visual C# project files (for example, C:\WSC4C\APPS).

In order to call WSC functions from your Visual C# programs, do the following to your existing C# source code:

Add the contents of file `wsc_funs.cs` to source code after

```
public class wsc : System.Windows.Forms.Form
```

Add the constants from `wsc_cons.cs` to your program as they are needed.

Look at the `cs_vers` program in the APPS directory for an example.

Note that if pointers are to be passed to Visual C# programs, the C# programs must be compiled in "unsafe" mode. This is necessary only when calling those functions (such as **SioPuts**, **SioGets**, or **SioWinError**) that pass pointers.

3.3 Microsoft C/C++ (16-bit only)

Support for Win16 was dropped beginning with version 5.2. Version 5.1 is still available (free when purchasing the current version) for those wanting Win16 support.

3.4 Borland C/C++

Borland C/C++ version 5.0 programs can be compiled from either the command line (using makefiles ending with "._b_") or from within the Borland development environment using Borland v5.0 or above.

Borland C/C++ Version 5.5 (which can be downloaded from www.borland.com) is a free Win32 console mode compiler (no IDE). Makefiles for BC v5.5 end with "._i_", and (like Borland C++ Builder) use ILINK32 rather than TLINK32. Be careful with linker response files (*.RSP) -- they must NOT end with a carriage return / line feed!

Borland programs always link with WSC32BC5.LIB.

3.4.1 Borland Command Line Makefiles

Programs can be compiled using command line makefiles. All Borland Win32 command line makefiles end with "`32._b_`" (or "`32._i_`" for Borland 5.5). To compile using a makefile, use the Borland MAKE utility. For example,

```
MAKE -f SIMPLE(LIB)32._B_
```

The file, *Makefiles(Borland50).zip*, contains the Borland C/C++ 5.0 command line makefiles.

the file, *Makefiles(Borland55).zip*, contains the Borland C/C++ 5.5 command line makefiles.

3.4.2 Borland IDE

To create a new project, first turn off LINKER case sensitivities: Choose "Options", "Projects", "Linker", "General". Turn off the "case sensitive link" and "case sensitive exports and imports" boxes.

Next, choose "Files", then "New Project". Use the INS (Insert) key to pop up a dialog box into which the project file names are entered. Lastly, add WSC32BCB.LIB to your project. WSC32BCB.LIB can also be created from WSC32.DLL using the Borland IMPLIB utility:

```
IMPLIB WSC32BCB.LIB WSC32.DLL
```

Select "GUI" or "Console" for the "Target Model:". Only "Static" or "Dynamic" should be checked for "Standard Libraries:"

NOTE1: If, after linking in the IDE, you get unresolved external references to the library functions in which each function name is all upper case, then you have NOT turned off case sensitivity as described above.

NOTE2: If you get errors compiling the windows header file "WINDOWS.H", turn on "Borland Extensions" in "Options", "Project", "Compiler", "Source".

3.5 Turbo C/C++ for Windows

Borland Turbo C/C++ for Windows does not have command line tools, so all programs must be compiled from the Turbo C/C++ integrated environment.

Follow the same directions as above (Borland IDE), except that the "Target Model:" can be any listed.

3.6 Borland C++ Builder

Borland C++ Builder does not have command line tools, so all programs must be compiled from the Borland C++ Builder integrated environment.

To load the SIMPLE_PRJ example project, Choose "File" / "Open Project" on the menu bar. Load SIMPLE _PRJ.BPR. Then, choose "Build All" from "Project" to create the executable.

Note that WSC32BCB.LIB is the LIB file used with Borland C++ Builder. WSC32BCB .LIB can be created from WSC32.DLL by using the Borland IMPLIB program:

```
IMPLIB WSC32BCB.LIB WSC32.DLL
```

3.7 Watcom C/C++

Watcom C/C++ programs can be compiled from either the command line or from within the Watcom development environment.

Watcom v10.5 and v10.6 do not recognize the "declspec" keyword.

3.7.1 Watcom Command Line Makefiles

Win32 programs can be compiled using command line makefiles. All Watcom command line makefiles end with "32._w_" for Win32 makefiles. To compile using a makefile, use the Watcom WMAKE utility. For example,

```
WMAKE -f SIMPLE(LIB)32._W_
```

Win32 programs can also be compiled using command line batch files. See SIMPLE$.BAT for an example of a console mode command line batch file and SIMPLE$.BAT for an example of a GUI mode command line batch file. To run these command line batch files from the command line, type

```
SIMPLE32
```

The file *Makefiles(Watcom11).zip* contains the WATCOM 11 command line makefiles for most of the example programs.

3.7.2 Watcom IDE

To create a new project, choose "File", then "New Project". Enter the project name and then choose Win32 as the target. Use the INS (Insert) key to pop up a dialog box into which the project file names are entered.

Select "Options" from the main window, then "C Compiler Switches", then "10". Memory Models and Processor Switches". Check "80386 Stack based calling [-3s]", then check "32-bit Flat model [-mf]".

3.8 LCC C/C++

The LCC-WIN32 and LCC-WIN-64 compilers were developed by the University of Virginia. See http://www.cs.virginia.edu/~lcc-win32/)

The file *Makefiles(lcc).zip* contains the LCC command line makefiles for most of the example programs.

3.9 MinGW C

MinGW (Minimalist GNU for Windows) is part of the GNU Compiler Collection (GCC), and GNU Binutils, for use in the development of native Microsoft Windows applications. See http://www.mingw.org.

The file *Makefiles(GCC).zip* contains the MinGW (GCC) project command line makefiles for most of the example programs.

Note: The GCC make executable (make.exe) is used, which requires libiconv2.dll and libintl3.dll

3.10 Digital Mars C

The Digital Mars Win32 C compiler is distributed by Digital Mars. See http://www.digitalmars.com.

The file *Makefiles(mars).zip* contains the Digital Mars command line makefiles for most of the example programs.

4 Compiling Programs

4.1 Compiling WSC Source Code

This section applies only to those who have purchased ANSII C source code for the WSC DLLs.

WSC32.DLL and WSC64.DLL have been compiled using Microsoft Visual C/C++, and are callable from applications written using Microsoft, Borland, or Watcom compilers. If you recompile WSC32.C using Borland or Watcom compilers, then the resulting DLL can only be used by applications compiled with the same compiler, unless the "_stdcall" and "_declspec" keywords are specified.

Microsoft Visual C++ is also used to create WSC32.OBJ and WSC64.OBJ (for static linking).

For Microsoft C, type:

```
NMAKE -f WSC32._M_
NMAKE -f WSC64._M_
```

For Borland C, type:

```
MAKE -f WSC32._B_
```

For Watcom C, type:

```
WMAKE -f WSC32._W_
```

Alternatively, WSC32.C can be included in a project (along with MSC-VS.C and MSC-STR.C) like any other C file. Before compiling, define the symbol STATIC_LIBRARY.

WSC64.DLL has been compiled using Microsoft Visual Studio 2008, and is callable from 64-bit applications programs.

4.2 Compiling Example Programs

The example programs can be compiled by using either the command line compiler or the compiler integrated development environment (IDE). Most compiler vendors provide both IDE and command line tools, although some compilers are command line only (Borland C/C++ 5.5 and LCC-WIN32) or IDE only (Borland C/C++ Builder).

Refer also to Section 3.0, "Supported Compilers".

4.2.1 Compiling Using Visual C++ (VC v4.0, v5.0, and v6.0)

Microsoft Visual C++ (v4.0, v5.0, v6.0) compiles only 32-bit programs.

Programs can be compiled with Microsoft Visual C++ using either Developer Studio / Microsoft Visual Studio or the command line compiler. Project makefiles (files ending in ".MAK") are provided for Developer Studio / Visual Studio. Command line makefiles (files ending in "._M_") are provided for use with the command line compiler (e.g.: `NMAKE -f SIMPLE(LIB)32._M_)`.

4.2.2 Compiling Using 16-bit Visual C++ (VC v1.52)

Support for Win16 was dropped beginning with version 5.2. Version 5.1 is still available (free when purchasing the current version) for those wanting Win16 support.

4.2.3 Compiling Using Microsoft Visual Studio C++ .Net

Microsoft Visual C++ .NET compiles only 32-bit programs.

Programs can be compiled with Microsoft Visual C++ .NET using either Visual Studio .NET or the command line compiler. Project files (files ending in ".VCPROJ") are provided for Visual Studio (eg: `VC_SIMPLE.VCPROJ`).

Command line makefiles (files ending in "._M_") are provided for use with the VC.NET command line compiler (e.g.: `NMAKE -f SIMPLE(LIB)32._M_)`.

4.2.4 Compiling Using Borland C/C++ 5.0

Borland C/C++ 5.0 can compile 32-bit programs.

Programs can be compiled with Borland C/C++ using either the Borland IDE or the command line compiler. Several Project files (files ending in ".IDE") are provided for the Borland IDE (unzip `BC50-IDE.ZIP)` and command line makefiles (files ending in "._B_") for the Borland command line compiler. For example (`MAKE -f SIMPLE(LIB)32._B_)`

4.2.5 Compiling Using Borland C/C++ 5.5

Borland C/C++ 5.5 is a command line compiler that can compile both 32-bit programs only. Command line makefiles (ending in "._I_") are provided. For example, (`MAKE -f SIMPLE(LIB)32._I_)`

4.2.6 Compiling Using Borland C++ Builder

Borland C++ Builder (BCB) is an IDE that features "drag and drop" forms building (like Delphi and Visual Basic).

BCB compiles 32-bit programs only. BCB project files end with the extension ".bpr". Load projects simple_prj.bpr and finder_prj.bpr.

4.2.7 Compiling Using Watcom 11

Programs can be compiled with Watcom 11 using either the Watcom IDE or the command line compiler. Several command line makefiles (files ending in "._W_") are provided for the Watcom command line compiler. For example,

```
WMAKE -f SIMPLE(LIB)32._W_
```

4.2.8 Compiling Using LCC

Several LCC command line makefiles (files ending in "._L_") are provided for the LCC command line compiler. For example,

```
MAKE -f SIMPLE(LIB)32._L_
```

Command line batch files (ending with "$LCC.BAT) are provided for several of the projects. For example, compile SIMPLE using LCC-WIN32 by typing SIMPLE$LCC.BAT on the command line.

4.2.9 Compiling Using MinGW C

Several GCC command line makefiles (files ending in "._G_") are provided for the GCC command line compiler. For example,

```
MAKE -f SIMPLE(LIB)32._G_
```

Command line batch files (ending with "$GCC.BAT) are provided for several of the projects. For example, compile SIMPLE using MinGW GCC by typing SIMPLE$GCC.BAT on the command line.

5 Example Programs

Some of the example programs are written in GUI mode (WSCVER, MFC_PGM, TERM, SELFTEST, SIMPLE), although most are written in Win32/Win64 console mode. This was done in order to provide the clearest possible code, without the complication and complexity of GUI code. All console mode programs can be converted to GUI mode by adding the necessary Windows interface code. Example programs are located in the \APPS directory.

Microsoft project files are classified as:

- `*.mak` Microsoft VC 4.0, 5.0, 6.0 project files (Win32).
- `*.dsp` Microsoft VC 6.0 project files (Win32 only).
- `*.vcproj` Microsoft Visual Studio project files (Win32 only).
- `VS2008(32bit).vcproj` Microsoft Visual Studio 2008 project files (Win32).
- `VS2008(64bit).vcproj` Microsoft Visual Studio 2008 project files (Win64).
- `VS2010(32bit).vcxproj` Microsoft Visual Studio 2010 project files (Win32).
- `VS2010(64bit).vcxproj` Microsoft Visual Studio 2010 project files (Win64).
- `VS2012(32bit).vcxproj` Microsoft Visual Studio 2012 project files (Win32).
- `VS2012(64bit).vcxproj` Microsoft Visual Studio 2012 project files (Win64).
- `VS2013(32bit).vcxproj` Microsoft Visual Studio 2013 project files (Win32).
- `VS2013(64bit).vcxproj` Microsoft Visual Studio 2013 project files (Win64).
- `VS2015(32bit).vcxproj` Microsoft Visual Studio 2015 project files (Win32).
- `VS2015(64bit).vcxproj` Microsoft Visual Studio 2015 project files (Win64).
- `*.bpr` Borland C++ Builder project file.

Command line makefiles are classified as:

- `*._M_` Microsoft C/C++ makefiles.
- `*._B_` Borland C/C++ 5.0 makefiles.
- `*._I_` Borland C/C++ 5.5 makefiles.
- `*._W_` Watcom C/C++ makefiles.
- `*._L_` LCC C makefiles.
- `*._D_` Digital Mars C makefiles.

Microsoft Visual C++ Developer Studio files end in ".MAK" and can be loaded with "Open Workspace" and include: simple.mak, term.mak, console.mak, event.mak, rs485.mak, xms.mak, xmr.mak, yms.mak, ymr.mak, finder.mak, echoport.mak, host.mak, device.mak, wscver.mak, dial.mak, & hello.mak.

Other files ending with .MAK (and .BPR) include:

- `MFCPGM32.MAK` Microsoft Foundations Class (MFC) makefile (Win32).
- `SIMPLE_PRJ.MAK` Borland C++ Builder makefile (BCB version 1 through 3).
- `SIMPLE_PRJ.BPR` Borland C++ Builder makefile (BCB version 4 and above).

NOTE: Unfortunately, the extension “.MAK” is used by computer manufacturers for both makefiles and project files. Further, a makefile that works with one compiler will not usually work with another.

5.1 WSCVER

WSCVER (WSC Version) is a GUI program that displays the WSC library version number. Its purpose is to display the WSC version, build, and registration string as well as to verify that WSC64.DLL (64-bit apps) or WSC32.DLL (32-bit apps) is being found and loaded by Windows.

Makefiles & project files are:

```
Microsoft 4.0-6.0 (32-bit) : wscver32._m_ makefile (unzip microsoft60.zip)
Microsoft 4.0-6.0 (32-bit) : wscver.mak   project file
Microsoft 6.0     (32-bit) : wscver.dsp   project file

Borland   5.0     (32-bit) : wscver32._b_ makefile (unzip Makefiles(Borland50).zip)
Borland   5.5     (32-bit) : wscver32._i_ makefile (unzip Makefiles(Borland55).zip)
Watcom   11.0     (32-bit) : wscver32._w_ makefile (unzip Makefiles(Watcom11).zip)
LCC-WIN32         (32-bit) : wscver$lcc.bat  batch (unzip Makefiles(LCC).zip)
MinGW GCC         (32-bit) : wscver$gcc.bat  batch (unzip Makefiles(GCC).zip)
```

VC_VERSION is the Visual Studio (C++ NET) version of WSCVER.

```
Visual Studio 2003/2005 (32-bit) : vc_version.vcproj
Visual Studio 2008      (32-bit) : unzip vs2008(32bit).vcproj
Visual Studio 2008      (64-bit) : unzip vs2008(64bit).vcproj
Visual Studio 2010      (32-bit) : unzip vs2010(32bit).vcxproj
Visual Studio 2010      (64-bit) : unzip vs2010(64bit).vcxproj
Visual Studio 2012      (32-bit) : unzip vs2012(32bit).vcxproj
Visual Studio 2012      (64-bit) : unzip vs2012(64bit).vcxproj
Visual Studio 2013      (32-bit) : unzip vs2013(32bit).vcxproj
Visual Studio 2013      (64-bit) : unzip vs2013(64bit).vcxproj
Visual Studio 2015      (32-bit) : unzip vs2015(32bit).vcxproj
Visual Studio 2015      (64-bit) : unzip vs2015(64bit).vcxproj
```

CS_VERS is the Visual Studio C# version of WSCVER / VC_VERSION.

```
Visual Studio C# 2003/2005/2008 (32-bit) : cs_vers.csproj
```

5.2 SIMPLE

SIMPLE is a very straightforward GUI mode terminal communications program using WSC4C. Everything typed on the keyboard is sent to the serial port and everything incoming from the serial port is displayed on the screen.

The easiest way to test SIMPLE is to connect to a modem. Typing "AT" should result in an "OK" being displayed.

Makefiles & project files are:

```
Microsoft 4.0-6.0 (32-bit) : simple32._m_ makefile (unzip microsoft60.zip)
Microsoft 4.0-6.0 (32-bit) : simple.mak   project file
Microsoft 6.0     (32-bit) : simple.dsp   project file
Borland   5.0     (32-bit) : simple32._b_ makefile (unzip Makefiles(Borland50).zip)
Borland   5.5     (32-bit) : simple32._i_ makefile (unzip Makefiles(Borland55).zip)
Watcom   11.0     (32-bit) : simple32._w_ makefile (unzip Makefiles(Watcom11).zip)
LCC-WIN32         (32-bit) : simple$lcc.bat  batch (unzip Makefiles(LCC).zip)
MinGW GCC         (32-bit) : simple$gcc.bat  batch (unzip Makefiles(GCC).zip)
Borland C++ Bld   (32-bit) : simple_prj.bpr project file
```

VC_SIMPLE is the Visual Studio (C++ NET) version of SIMPLE.

```
Visual Studio 2003/2005 (32-bit) : vc_simple.vcproj
Visual Studio 2008      (32-bit) : unzip vs2008(32bit).vcproj
Visual Studio 2008      (64-bit) : unzip vs2008(64bit).vcproj
Visual Studio 2010      (32-bit) : unzip vs2010(32bit).vcxproj
Visual Studio 2010      (64-bit) : unzip vs2010(64bit).vcxproj
Visual Studio 2012      (32-bit) : unzip vs2012(32bit).vcxproj
Visual Studio 2012      (64-bit) : unzip vs2012(64bit).vcxproj
Visual Studio 2013      (32-bit) : unzip vs2013(32bit).vcxproj
Visual Studio 2013      (64-bit) : unzip vs2013(64bit).vcxproj
Visual Studio 2015      (32-bit) : unzip vs2015(32bit).vcxproj
Visual Studio 2015      (64-bit) : unzip vs2015(64bit).vcxproj
```

5.3 CONSOLE

The CONSOLE program is a console mode program similar to SIMPLE. It also demonstrates how to acquire and manipulate the standard input handle so that a "KeyPress" function can be implemented.

CONSOLE takes the port and baud rate as arguments. For example, to start CONSOLE on COM1 at 38,400 baud:

```
CONSOLE 1 38400
```

Makefiles & project files are:

```
Microsoft 4.0-6.0 (32-bit) : console32._m_ makefile (unzip microsoft60.zip)
Microsoft 4.0-6.0 (32-bit) : console.mak   project file
Microsoft 6.0     (32-bit) : console.dsp   project file
Borland   5.0     (32-bit) : console32._b_ makefile (unzip Makefiles(Borland50).zip)
Borland   5.5     (32-bit) : console32._i_ makefile (unzip Makefiles(Borland55).zip)
Watcom   11.0     (32-bit) : console32._w_ makefile (unzip Makefiles(Watcom11).zip)
LCC-WIN32         (32-bit) : console$lcc.bat  makefile (unzip Makefiles(LCC).zip)
MinGW C           (32-bit) : console$gcc.bat  makefile (unzip Makefiles(GCC).zip)
```

VC_CONSOLE.CPP if the Visual Studio (C++ NET) version of CONSOLE

```
Visual Studio 2003/2005 (32-bit) : vc_console.vcproj
Visual Studio 2008      (32-bit) : unzip vs2008(32bit).vcproj
Visual Studio 2008      (64-bit) : unzip vs2008(64bit).vcproj
Visual Studio 2010      (32-bit) : unzip vs2010(32bit).vcxproj
Visual Studio 2010      (64-bit) : unzip vs2010(64bit).vcxproj
Visual Studio 2012      (32-bit) : unzip vs2012(32bit).vcxproj
Visual Studio 2012      (64-bit) : unzip vs2012(64bit).vcxproj
Visual Studio 2013      (32-bit) : unzip vs2012(32bit).vcxproj
Visual Studio 2013      (64-bit) : unzip vs2013(64bit).vcxproj
Visual Studio 2015      (32-bit) : unzip vs2015(32bit).vcxproj
Visual Studio 2015      (64-bit) : unzip vs2015(64bit).vcxproj
```

5.4 RS485

The RS485 example console mode program operates like SIMPLE, except that it assumes an RS485 port. RTS is set before transmitting data, and cleared after the last bit of the last byte has been sent.

Makefiles & project files are:

```
Microsoft 4.0-6.0 (32-bit) : RS485._m_   makefile (unzip microsoft60.zip)
Microsoft 4.0-6.0 (32-bit) : RS485.mak   project file
Microsoft 6.0     (32-bit) : RS485.dsp   project file
Borland   5.0     (32-bit) : RS485._b_   makefile (unzip Makefiles(Borland50).zip)
Borland   5.5     (32-bit) : RS485._i_   makefile (unzip Makefiles(Borland55).zip)
Watcom   11.0     (32-bit) : RS485._w_   makefile (unzip Makefiles(Watcom11).zip)
```

5.5 SELFTEST

SELFTEST is a GUI mode program that performs serial port I/O functionality testing using a loopback adapter. Refer to LOOPBACK.TXT for an explanation of how to make a loopback adapter (without tools!).

Makefiles & project files are:

```
Microsoft 4.0-6.0 (32-bit) : self32._m_ makefile (unzip microsoft60.zip)
Microsoft 4.0-6.0 (32-bit) : selftest.mak   project file
Microsoft 6.0     (32-bit) : selftest.dsp   project file
Borland   5.0     (32-bit) : self32._b_ makefile (unzip Makefiles(Borland50).zip)
Borland   5.5     (32-bit) : self32._i_ makefile (unzip Makefiles(Borland55).zip)
Watcom   11.0     (32-bit) : self32._w_ makefile (unzip Makefiles(Watcom11).zip)
```

5.6 TERM

TERM is a simple GUI mode ANSI terminal emulator suitable downloading or uploading files using XMODEM or YMODEM. The TERM program uses MIO32.DLL for modem control commands, and XYDRV32.DLL for XMODEM & YMODEM file transfer.

Selecting "Dial" from the menu bar will result in a pop-up dialog requesting the phone number to dial. Once entered, the number is dialed, and the program will wait for up to 60 seconds for the "CONNECT" string from the modem. This wait can be terminated at any time by choosing "BREAK" on the menu bar.

Once logged on, files can be uploaded or downloaded by selecting "Send" or "Receive" from the menu bar. To abort a file transfer, choose "BREAK" from the menu bar then type a series of Ctrl-X (^X) characters from the keyboard.

TERM uses ANSI.C, which provides ANSI terminal emulator support.

Makefiles & project files are:

```
Microsoft 4.0-6.0 (32-bit) : term32._m_ makefile (unzip microsoft60.zip)
Microsoft 4.0-6.0 (32-bit) : term.mak   project file
Microsoft 6.0     (32-bit) : term.dsp   project file
Borland   5.0     (32-bit) : term32._b_ makefile (unzip Makefiles(Borland50).zip)
Borland   5.5     (32-bit) : term32._i_ makefile (unzip Makefiles(Borland55).zip)
Watcom   11.0     (32-bit) : term32._w_ makefile (unzip Makefiles(Watcom11).zip)
LCC-WIN32         (32-bit) : term$lcc.bat  batch (unzip Makefiles(LCC).zip)
MinGW C           (32-bit) : term$gcc.bat  batch (unzip Makefiles(GCC).zip)
```

5.7 CALLBACK

The Callback example program demonstrates how to implement a WSC callback function. The code necessary to perform the callback is encapsulated in the SioCallback.c file.

Makefiles & project files are:

```
Microsoft 4.0-6.0 (32-bit) : CallBack._m_ makefile (unzip microsoft60.zip)
Microsoft 4.0-6.0 (32-bit) : CallBack.mak project file
Microsoft 6.0     (32-bit) : CallBack.dsp project file
Borland   5.0     (32-bit) : CallBack._b_ makefile (unzip Makefiles(Borland50).zip)
Borland   5.5     (32-bit) : CallBack._i_ makefile (unzip Makefiles(Borland55).zip)
Watcom   11.0     (32-bit) : CallBack._w_ makefile (unzip Makefiles(Watcom11).zip)
```

5.8 DEVICE

The DEVICE console mode program is designed for talking to an arbitrary serial device. Use this program as a guide when communicating with all serial devices other than modems and other computers.

Makefiles & project files are:

```
Microsoft 4.0-6.0 (32-bit) : Device._m_   makefile (unzip microsoft60.zip)
Microsoft 4.0-6.0 (32-bit) : Device.mak   project file
Microsoft 6.0     (32-bit) : Device.dsp   project file
Borland   5.0     (32-bit) : Device._b_   makefile (unzip Makefiles(Borland50).zip)
Borland   5.5     (32-bit) : Device._i_   makefile (unzip Makefiles(Borland55).zip)
Watcom   11.0     (32-bit) : Device._w_   makefile (unzip Makefiles(Watcom11).zip)
```

VC_DEVICE is the Visual Studio (C++ NET) version of DEVICE.

```
Visual Studio 2003/2005 (32-bit) : vc_device.vcproj
Visual Studio 2008      (32-bit) : unzip vs2008(32bit).vcproj
Visual Studio 2008      (64-bit) : unzip vs2008(64bit).vcproj
Visual Studio 2010      (32-bit) : unzip vs2010(32bit).vcxproj
Visual Studio 2010      (64-bit) : unzip vs2010(64bit).vcxproj
Visual Studio 2012      (32-bit) : unzip vs2012(32bit).vcxproj
Visual Studio 2012      (64-bit) : unzip vs2012(64bit).vcxproj
Visual Studio 2013      (32-bit) : unzip vs2012(32bit).vcxproj
Visual Studio 2013      (64-bit) : unzip vs2013(64bit).vcxproj
Visual Studio 2015      (32-bit) : unzip vs2015(32bit).vcxproj
Visual Studio 2015      (64-bit) : unzip vs2015(64bit).vcxproj
```

5.9 DIAL

The DIAL console mode example program uses the MIO module to dial up using the port, baud rate, and phone number specified by the user on the command line. For example

```
DIAL 1 38400 "1,256,555,1212"
```

Makefiles & project files are:

```
Microsoft 4.0-6.0 (32-bit) : Dial32._m_   makefile (unzip microsoft60.zip)
Microsoft 4.0-6.0 (32-bit) : Dial.mak     project file
Microsoft 6.0     (32-bit) : Dial.dsp     project file
Borland   5.0     (32-bit) : Dial32._b_   makefile (unzip Makefiles(Borland50).zip)
Borland   5.5     (32-bit) : Dial32._i_   makefile (unzip Makefiles(Borland55).zip)
Watcom   11.0     (32-bit) : Dial32._w_   makefile (unzip Makefiles(Watcom11).zip)
```

5.10 EchoPort

The EchoPort console mode program demonstrates the use of threads. Connect COM1 and/or COM2 to another computer or serial device using a NULL modem cable. EchoPort will echo back to the remote anything that it receives on the serial port. EchoPort takes no arguments.

Makefiles & project files are:

```
Microsoft 4.0-6.0 (32-bit) : EchoPort._m_ makefile (unzip microsoft60.zip)
Microsoft 4.0-6.0 (32-bit) : EchoPort.mak project file
Microsoft 6.0     (32-bit) : EchoPort.dsp project file
Borland   5.0     (32-bit) : EchoPort._b_ makefile (unzip Makefiles(Borland50).zip)
Borland   5.5     (32-bit) : EchoPort._i_ makefile (unzip Makefiles(Borland55).zip)
Watcom   11.0     (32-bit) : EchoPort._w_ makefile (unzip Makefiles(Watcom11).zip)
```

5.11 Event, EventC, EventW

The EVENT console mode program demonstrates the use of **SioEvent** in efficiently waiting for serial input. **SioEvent** blocks until the requested event occurs.

Also see the EventC example program that uses **SioEventChar** and the EventW example program that uses **SioEventWait**. Refer to Section 2.9 for information on the SioEvent functions.

Makefiles & project files are:

```
Microsoft 4.0-6.0 (32-bit) : Event32._m_ makefile (unzip microsoft60.zip)
Microsoft 4.0-6.0 (32-bit) : Event.mak   project file
Microsoft 6.0     (32-bit) : Event.dsp   project file
Borland   5.0     (32-bit) : Event32._b_ makefile (unzip Makefiles(Borland50).zip)
Borland   5.5     (32-bit) : Event32._i_ makefile (unzip Makefiles(Borland55).zip)
Watcom   11.0     (32-bit) : Event32._w_ makefile (unzip Makefiles(Watcom11).zip)
LCC-WIN32         (32-bit) : Event$lcc.bat  makefile (unzip Makefiles(LCC).zip)
MinGW C           (32-bit) : Event$gcc.bat  makefile (unzip Makefiles(GCC).zip)

Microsoft 4.0-6.0 (32-bit) : EventC._m_  makefile (unzip microsoft60.zip)
Microsoft 4.0-6.0 (32-bit) : EventC.mak  project file
Microsoft 6.0     (32-bit) : EventC.dsp  project file
Borland   5.0     (32-bit) : EventC._b   makefile (unzip Makefiles(Borland50).zip)
Borland   5.5     (32-bit) : EventC._i_  makefile (unzip Makefiles(Borland55).zip)
Watcom   11.0     (32-bit) : EventC._w_  makefile (unzip Makefiles(Watcom11).zip)

Microsoft 4.0-6.0 (32-bit) : EventW._m_   makefile (unzip microsoft60.zip)
Microsoft 4.0-6.0 (32-bit) : EventW.mak   project file
Microsoft 6.0     (32-bit) : EventW.dsp   project file
Borland   5.0     (32-bit) : EventW._b_   makefile (unzip Makefiles(Borland50).zip)
Borland   5.5     (32-bit) : EventW._i_   makefile (unzip Makefiles(Borland55).zip)
Watcom   11.0     (32-bit) : EventW._w_   makefile (unzip Makefiles(Watcom11).zip)
```

VC_EVENT is the Visual Studio (C++ NET) version of EVENT.

```
Visual Studio 2003/2005 (32-bit) : vc_event.vcproj
Visual Studio 2008      (32-bit) : unzip vs2008(32bit).vcproj
Visual Studio 2008      (64-bit) : unzip vs2008(64bit).vcproj
Visual Studio 2010      (32-bit) : unzip vs2010(32bit).vcxproj
Visual Studio 2010      (64-bit) : unzip vs2010(64bit).vcxproj
Visual Studio 2012      (32-bit) : unzip vs2012(32bit).vcxproj
Visual Studio 2012      (64-bit) : unzip vs2012(64bit).vcxproj
Visual Studio 2013      (32-bit) : unzip vs2012(32bit).vcxproj
Visual Studio 2013      (64-bit) : unzip vs2013(64bit).vcxproj
Visual Studio 2015      (32-bit) : unzip vs2015(32bit).vcxproj
Visual Studio 2015      (64-bit) : unzip vs2015(64bit).vcxproj
```

5.12 FINDER

The FINDER program is a console mode program that searches for a connected modem. The modem must be turned on. FINDER takes no arguments.

Makefiles & project files are:

```
Microsoft 4.0-6.0 (32-bit) : finder32._m_ makefile (unzip microsoft60.zip)
Microsoft 4.0-6.0 (32-bit) : finder.mak   project file
Microsoft 6.0     (32-bit) : finder.dsp   project file
Borland   5.0     (32-bit) : finder32._b_ makefile (unzip Makefiles(Borland50).zip)
Borland   5.5     (32-bit) : finder32._i_ makefile (unzip Makefiles(Borland55).zip)
Watcom   11.0     (32-bit) : finder32._w_ makefile (unzip Makefiles(Watcom11).zip)
LCC-WIN32         (32-bit) : finder$lcc.bat cmd file (unzip Makefiles(LCC).zip)
MinGW GCC         (32-bit) : finder$gcc.bat cmd file (unzip Makefiles(GCC).zip)
Borland C++ Bld   (32-bit) : finder_prj.bpr project file
```

VC_FINDER is the Visual Studio (C++ NET) version of FINDER.

```
 Visual Studio 2003/2005 (32-bit) : vc_finder.vcproj
Visual Studio 2008      (32-bit) : unzip vs2008(32bit).vcproj
Visual Studio 2008      (64-bit) : unzip vs2008(64bit).vcproj
Visual Studio 2010      (32-bit) : unzip vs2010(32bit).vcxproj
Visual Studio 2010      (64-bit) : unzip vs2010(64bit).vcxproj
Visual Studio 2012      (32-bit) : unzip vs2012(32bit).vcxproj
Visual Studio 2012      (64-bit) : unzip vs2012(64bit).vcxproj
Visual Studio 2013      (32-bit) : unzip vs2012(32bit).vcxproj
Visual Studio 2013      (64-bit) : unzip vs2013(64bit).vcxproj
Visual Studio 2015      (32-bit) : unzip vs2015(32bit).vcxproj
Visual Studio 2015      (64-bit) : unzip vs2015(64bit).vcxproj
```

5.14 HELLO

The Hello.cpp console mode example program demonstrates how to use the fSio C++ class.

```
Microsoft 4.0-6.0 (32-bit) : Hello._m_  makefile (unzip microsoft60.zip)
Microsoft 4.0-6.0 (32-bit) : Hello.mak  project file
Microsoft 6.0     (32-bit) : Hello.dsp  project file
Borland   5.0     (32-bit) : Hello._b_  makefile (unzip Makefiles(Borland50).zip)
Borland   5.5     (32-bit) : Hello._i_  makefile (unzip Makefiles(Borland55).zip)
Watcom   11.0     (32-bit) : Hello._w_  makefile (unzip Makefiles(Watcom11).zip)
```

5.15 HOST

The HOST console mode program answers an incoming modem call. It functions as a very simple BBS (Bulletin Board System) program. HOST uses both WSC32.DLL and MIO32.DLL. Start HOST from the command line passing the port and baud rate. For example,

```
HOST 1 19200
```

Makefiles & project files are:

```
Microsoft 4.0-6.0 (32-bit) : Host32._m_   makefile (unzip microsoft60.zip)
Microsoft 4.0-6.0 (32-bit) : Host.mak     project file
Microsoft 6.0     (32-bit) : Host.dsp     project file
Borland   5.0     (32-bit) : Host32._b_   makefile (unzip Makefiles(Borland50).zip)
Borland   5.5     (32-bit) : Host32._i_   makefile (unzip Makefiles(Borland55).zip)
Watcom   11.0     (32-bit) : Host32._w_   makefile (unzip Makefiles(Watcom11).zip)
```

5.13 LISTER

The LISTER program lists all serial ports. LISTER takes no arguments.

Makefiles & project files are:

```
Microsoft 4.0-6.0 (32-bit) : lister32._m_ makefile (unzip microsoft60.zip)
Microsoft 4.0-6.0 (32-bit) : lister.mak   project file
Microsoft 6.0     (32-bit) : lister.dsp   project file
Borland   5.0     (32-bit) : lister 32._b_ makefile (unzip Makefiles(Borland50).zip)
Borland   5.5     (32-bit) : lister 32._i_ makefile (unzip Makefiles(Borland55).zip)
Watcom   11.0     (32-bit) : lister32._w_  makefile (unzip Makefiles(Watcom11).zip)
LCC-WIN32         (32-bit) : lister $lcc.bat cmd file (unzip Makefiles(LCC).zip)
MinGW GCC         (32-bit) : lister $gcc.bat cmd file (unzip Makefiles(GCC).zip)
Borland C++ Bld   (32-bit) : lister_prj.bpr project file
```

VC_LISTER is the Visual Studio (C++ NET) version of LISTER.

```
Visual Studio 2003/2005 (32-bit) : vc_lister.vcproj
Visual Studio 2008      (32-bit) : unzip vs2008(32bit).vcproj
Visual Studio 2008      (64-bit) : unzip vs2008(64bit).vcproj
Visual Studio 2010      (32-bit) : unzip vs2010(32bit).vcxproj
Visual Studio 2010      (64-bit) : unzip vs2010(64bit).vcxproj
Visual Studio 2012      (32-bit) : unzip vs2012(32bit).vcxproj
Visual Studio 2012      (64-bit) : unzip vs2012(64bit).vcxproj
Visual Studio 2013      (32-bit) : unzip vs2012(32bit).vcxproj
Visual Studio 2013      (64-bit) : unzip vs2013(64bit).vcxproj
Visual Studio 2015      (32-bit) : unzip vs2015(32bit).vcxproj
Visual Studio 2015      (64-bit) : unzip vs2015(64bit).vcxproj
```

5.22 ProXR

The ProXR console mode example program demonstrates how to control the ProXR relays boards manufactured by ControlAnything.com

Makefiles & project files are:

```
Microsoft 4.0-6.0 (32-bit) : ProXR._m_   makefile (unzip microsoft60.zip)
Microsoft 4.0-6.0 (32-bit) : ProXR.mak   project file
Microsoft 6.0     (32-bit) : ProXR.dsp   project file
Borland   5.0     (32-bit) : ProXR._b_   makefile (unzip Makefiles(Borland50).zip)
Borland   5.5     (32-bit) : ProXR._i_   makefile (unzip Makefiles(Borland55).zip)
```

VC_ProXR is the Visual Studio (C++ NET) version of ProXR.

```
Visual Studio 2003/2005 (32-bit) : vc_ProXR.vcproj
Visual Studio 2008      (32-bit) : unzip vs2008(32bit).vcproj
Visual Studio 2008      (64-bit) : unzip vs2008(64bit).vcproj
Visual Studio 2010      (32-bit) : unzip vs2010(32bit).vcxproj
Visual Studio 2010      (64-bit) : unzip vs2010(64bit).vcxproj
Visual Studio 2012      (32-bit) : unzip vs2012(32bit).vcxproj
Visual Studio 2012      (64-bit) : unzip vs2012(64bit).vcxproj
Visual Studio 2013      (32-bit) : unzip vs2012(32bit).vcxproj
Visual Studio 2013      (64-bit) : unzip vs2013(64bit).vcxproj
Visual Studio 2015      (32-bit) : unzip vs2015(32bit).vcxproj
Visual Studio 2015      (64-bit) : unzip vs2015(64bit).vcxproj
```

5.16 XMS and XMR

XMS (XMODEM Send) and XMR (XMODEM Receive) are console mode programs that send and receive files using the XMODEM protocol.

Makefiles & project files are:

```
Microsoft 4.0-6.0 (32-bit) : XMS32._m_   makefile (unzip microsoft60.zip)
Microsoft 4.0-6.0 (32-bit) : XMS.mak     project file
Microsoft 6.0     (32-bit) : XMS.dsp     project file
Borland   5.0     (32-bit) : XMS32._b_   makefile (unzip Makefiles(Borland50).zip)
Borland   5.5     (32-bit) : XMS32._i_   makefile (unzip Makefiles(Borland55).zip)
Watcom   11.0     (32-bit) : XMS32._w_   makefile (unzip Makefiles(Watcom11).zip)

Microsoft 4.0-6.0 (32-bit) : XMR32._m_   makefile (unzip microsoft60.zip)
Microsoft 4.0-6.0 (32-bit) : XMR.mak     project file
Microsoft 6.0     (32-bit) : XMR.dsp     project file
Borland   5.0     (32-bit) : XMR32._b_   makefile (unzip Makefiles(Borland50).zip)
Borland   5.5     (32-bit) : XMR32._i_   makefile (unzip Makefiles(Borland55).zip)
Watcom   11.0     (32-bit) : XMR32._w_   makefile (unzip Makefiles(Watcom11).zip)
```

VC_XMS and VC_XMR.CPP are the Visual Studio (C++ NET) versions of XMS and XMR.

```
Visual Studio 2003/2005 (32-bit) : vc_xms.vcproj
Visual Studio 2008      (32-bit) : unzip vs2008(32bit).vcproj
Visual Studio 2008      (64-bit) : unzip vs2008(64bit).vcproj
Visual Studio 2010      (32-bit) : unzip vs2010(32bit).vcxproj
Visual Studio 2010      (64-bit) : unzip vs2010(64bit).vcxproj
Visual Studio 2012      (32-bit) : unzip vs2012(32bit).vcxproj
Visual Studio 2012      (64-bit) : unzip vs2012(64bit).vcxproj

Visual Studio 2003/2005 (32-bit) : vc_xmr.vcproj
Visual Studio 2008      (32-bit) : unzip vs2008(32bit).vcproj
Visual Studio 2008      (64-bit) : unzip vs2008(64bit).vcproj
Visual Studio 2010      (32-bit) : unzip vs2010(32bit).vcxproj
Visual Studio 2010      (64-bit) : unzip vs2010(64bit).vcxproj
Visual Studio 2012      (32-bit) : unzip vs2012(32bit).vcxproj
Visual Studio 2012      (64-bit) : unzip vs2012(64bit).vcxproj
Visual Studio 2013      (32-bit) : unzip vs2012(32bit).vcxproj
Visual Studio 2013      (64-bit) : unzip vs2013(64bit).vcxproj
Visual Studio 2015      (32-bit) : unzip vs2015(32bit).vcxproj
Visual Studio 2015      (64-bit) : unzip vs2015(64bit).vcxproj
```

5.17 YMS and YMR

YMS (YMODEM Send) and YMR (YMODEM Receive) are console mode programs that send and receive files using the YMODEM protocol.

Makefiles & project files are:

```
Microsoft 4.0-6.0 (32-bit) : YMS32._m_   makefile (unzip microsoft60.zip)
Microsoft 4.0-6.0 (32-bit) : YMS.mak     project file
Microsoft 6.0     (32-bit) : YMS.dsp     project file
Borland   5.0     (32-bit) : YMS32._b_   makefile (unzip Makefiles(Borland50).zip)
Borland   5.5     (32-bit) : YMS32._i_   makefile (unzip Makefiles(Borland55).zip)
Watcom   11.0     (32-bit) : YMS32._w_   makefile (unzip Makefiles(Watcom11).zip)

Microsoft 4.0-6.0 (32-bit) : YMR32._m_   makefile (unzip microsoft60.zip)
Microsoft 4.0-6.0 (32-bit) : YMR.mak     project file
Microsoft 6.0     (32-bit) : YMR.dsp     project file
Borland   5.0     (32-bit) : YMR32._b_   makefile (unzip Makefiles(Borland50).zip)
Borland   5.5     (32-bit) : YMR32._i_   makefile (unzip Makefiles(Borland55).zip)
Watcom   11.0     (32-bit) : YMR32._w_   makefile (unzip Makefiles(Watcom11).zip)
```

5.18 MFC_PGM

MFC_PGM is a Microsoft Foundation Class C++ (GUI mode) program similar to SIMPLE. The file NAFXCW.LIB may have to be copied from MFC\LIB on your Microsoft Visual C/C++ CD disProject files are:

```
Microsoft C++ (32-bit) : mfcpgm32.mak project file
```

5.19 CS_AT_OK

The CS_AT_OK example Visual C# program transmits the 3 byte character string "AT\n" on COM1 at 19200 baud. After sleeping a short period of time, the port is read for any response.

This example program demonstrates how to send and receive using SioPutc and SioGetc from a Visual C# application program.

Recall that if "AT\n" is sent to a modem where word response codes are enabled, the string "OK" (without the quotes) will be returned by the modem.

Note that the program must be compiled using the “/unsafe” Visual C# keyword.

Also refer to Section 3.2 “Microsoft Visual C#” above.

The Project file is:

```
Visual Studio C# 2003/2005/2008/2010/2012/2013/2015 (32-bit): cs_at_ok.csproj
```

5.20 CS_AT_OK_2

The CS_AT_OK example C# program is similar to CS_AT_OK, except that it uses the SioByteToShort and SioShortToByte functions for converting 8-bit Character Buffer to and from 16-bit Unicode ASCII.

The project file is:

```
Visual Studio C# 2003/2005/2008/2010/2012/2013/2015 (32-bit) : cs_at_ok_2.csproj
```

5.21 LoadLib

The LoadLib console mode example program demonstrates how to load WSC functions dynamically from a specified location.

Makefiles & project files are:

```
Microsoft 4.0-6.0 (32-bit) : LoadLib._m_  makefile (unzip microsoft60.zip)
Microsoft 4.0-6.0 (32-bit) : LoadLib.mak  project file
Microsoft 6.0     (32-bit) : LoadLib.dsp  project file
Borland   5.0     (32-bit) : LoadLib._b_  makefile (unzip Makefiles(Borland50).zip)
Borland   5.5     (32-bit) : LoadLib._i_  makefile (unzip Makefiles(Borland55).zip)
Watcom   11.0     (32-bit) : LoadLib._w_  makefile (unzip Makefiles(Watcom11).zip)
```

6 Additional Examples

We have other examples programs including programs for controlling voice modems (which use the Rockwell voice/data chip set), an example of handling continuously incoming data, and others. Please contact us if you need an example program not listed above.

7 Revision History

Version 1.0: September 6, 1996.

- The initial release of WSC4C.

Version 2.0: January 25, 1997.

- Includes Win32 libraries [Microsoft, Borland, Watcom].
- Added XMODEM & YMODEM DLL (XYDRIVER.DLL).
- Added TERM example program.

Version 2.1: June 2, 1997.

- Added ANSI terminal emulator support.
- Added "expires" option to SioInfo(). [Evaluation version ONLY].
- Added SioRead function.
- Added Borland C Builder example program.

Version 2.2: October 1, 1997.

- Fixed problem in XYDRV.C (renamed from XYDRIVER.C)
- Added xyGetFileName function to XYDRV.
- WSC4C supports up to 16 ports.
- WSC4C runs under Windows NT.
- Added Microsoft Foundation Class (MFC) Example.
- Added ASCII protocol.

Version 2.3: July 15, 1998.

- Multiple changes to XYDRV for added capability.
- SioTimer() function added to WSC16 and WSC32.
- FINDER example modem search program added.
- CONSOLE example console mode program added.
- ECHOPORT example multi-thread program added.
- SioBaud & SioParms can be called before SioReset.
- WSC32.C modified so that the number of supported ports can be easily increased.

Version 2.4: May 17, 1999.

- SioEvent function added (WIN32 only).
- New xyDriver code.
- New XMODEM example programs (XMS and XMR).
- New RS485 example program.
- Specify default DTR & RTS behavior with SioReset.

Version 3.0: July 12, 2000.
- Added SioMessage function.
- New WORD and HTML documentation.
- New example programs.

Version 3.1: April 19, 2001.
- RESETDEV Win API call not called (allows USB/serial converters).
- SioPutc and SioPuts return immediately.
- WSC32.C can be recompiled for Win/CE.

Version 3.2: July 17, 2002.

- Modified so can be compiled for Windows CE [USE_WIN_CE].
- Default for RESETDEV is "not called". SioDebug('R') to enable.
- SioGetc & SioGets zero unused bits (DataBits 5,6,7).
- Corrected problem with SioBaud(-1, BaudRateCode).
- SioDebug returns -1 if no match.
- Added SioDebug('W') toggle SioPuts wait for I/O completion.
- Added code to detect active threads & to close thread handles.
- Added USE_THREADS, so can compile version of WSC32.C without threads.
- Comm handle not saved in SioReset unless it is good.
- SioEvent returns mask that caused the event.
- Added SioInfo('B') to get build number.

Version 3.3: October 28, 2003.

- Can now order either with or without source code to the DLLs.
- Added SioSetInteger function to set port specific integer parameters.
- Added SioKeyCode function to pass the key code to the DLL.
- Added SioGetReg function to return the registration string.
- Added "Burst Size" parameter for setting the TX burst size.
- Added ability to signal blocked thread which was blocked by SioEvent.

Version 4.0: November 12, 2003.

- Added support for Microsoft VC.Net

Version 4.1: August 12, 2004

- Fixed problem with SioTxClear.
- Added overlapped I/O (for non-Win95) so can signal threads to exit w/o killing them.
- Increased default burst size to 256.
- SioFlow returns WSC_RANGE if cannot recognize parameter.
- Added support for Microsoft C#.
- Adjusted XModem/YModem timing for faster transfers.

Version 4.2: February 1, 2006.

- SioFlow returns 1 if OK.
- SioSetInteger(Port, 'S', 1) always forces SioEvent to unblock.
- Event mutex code added to EventThread() to prevent race conditions.
- Message box displays error if SioWinError(Buffer, 0) called.
- Major change in overlapped I/O
- Fixed problem: SioEvent returning wrong code.
- SioRxClear clears byte saved by SioUnGet.
- Number of supported ports increased to a maximum of 256.
- Added SioEventChar() and SioEventWait() functions.

Version 4.3: September 25, 2007.

- Fixed problem with SioTxQue returning wrong values.
- Changed SioParms so it checks the range of passed arguments.
- Port is verified in SioEventChar.
- SioStatus returns -1 if port is not functioning (USB/serial port disconnected).
- Added SioByteToShort and SioShortToByte (WSC32 only).
- LoadLib.c and Callback.c example programs added.

Version 4.4: January 19, 2009.

- Added SioSetTimeouts() function (sets TX and RX time-outs).

Version 5.0: November 16, 2009

- Added SioHexView() function.
- Supports 64-bits (WSC64.DLL).
- Added several Visual Studio examples.

Version 5.1: August 19, 2011

- Added SioRxWait() function.
- Added support for MinGW GCC compiler.
- Added support for Visual Studio 2010 C/C++ compiler.
- Added LISTER.C example program.

Version 5.2: July 2, 2012

- Added function SioQuiet()
- Added function SioWaitFor()
- Added example program ProXR.c
- Win16 support dropped, but available on request when ordering.

Version 5.3: November 4, 2013

- Added SioLRC() that computes the "longitudinal redundancy check" per ISO 1155.
- SioQuiet and SioWaitFor verify the passed port number.
- SioWaitFor verifies that the passed baud rate is > 0.
- SioSetInteger no longer requires an open port for global (all ports) parameters.
- Modified SioReset to make it more tolerant opening slow virtual ports.

Version 5.4: August 11, 2015

- Added makefiles for GCC, LCC, and Digital Mars C compilers.
- Added SioCRC16() that computes 16-bit CCITT CRC (polynomial 1021 hex).
- Added SioCRC32() that computes 32-bit CCITT CRC (polynomial 04C11DB7 hex).
- Added SioCountWait() that waits for a specified number of incoming bytes before returning.
- Added support for Visual Studio 2013 and Visual Studio 2015.